SECURE COMPUTER SYSTEMS: A REFINEMENT OF THE MATHEMATICAL MODEL

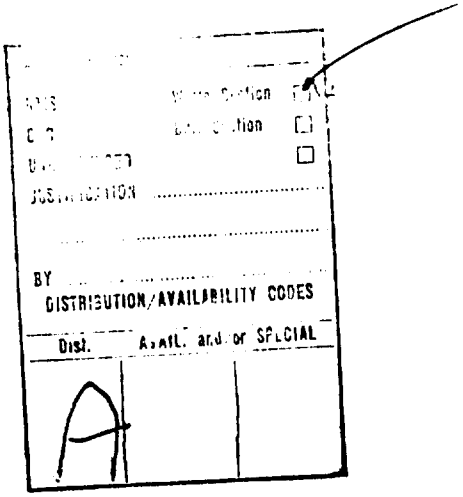
D. Elliott Bell

Mitre Corporation

Prepared for:

Electronic Systems Division

April 1974



REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

William R. Price

WILLIAM R. PRICE, 1Lt, USAF
Project Officer

Arthur E. Whitson

ARTHUR E. WHITSON, Lt Col, USAF
Chief, Techniques Division

Robert W. O'Keefe

ROBERT W. O'KEEFE, Col, USAF
Director, Information Systems Technology
Deputy for Command & Management Systems

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER ESD-TR-73-278, Vol. III	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SECURE COMPUTER SYSTEMS: A REFINEMENT OF THE MATHEMATICAL MODEL		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER MTR-2547, Vol. III
7 AUTHOR(s) D. E. Bell		8. CONTRACT OR GRANT NUMBER(S) F19628-73-C-0001
9 PERFORMING ORGANIZATION NAME AND ADDRESS The MITRE Corporation Box 208 Bedford, Mass. 01730		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project 522G
11 CONTROLLING OFFICE NAME AND ADDRESS Deputy for Command and Management Systems Electronic Systems Division, AFSC L. G. Hanscom Field, Bedford, Mass. 01730		12 REPORT DATE APRIL 1974 13. NUMBER OF PAGES 76
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

OBJECT STRUCTURE
CURRENT SECURITY LEVEL
ALTERED *-PROPERTY

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A model developed for the investigation of security in computer systems is refined in three major ways, incorporating an object structure, a notion of current security level, and an altered *-property. In addition, the various ramifications of classifying a control structure are explored. It is shown that security requirements can be fulfilled in a system using these refinements.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ACKNOWLEDGEMENTS

The author wishes to express his thanks to the many who have contributed so much to this report. I owe special thanks to W. L. Schiller for his observations which led to the results included in Section III. I am further endebted to E. L. Burke for his insights into security and the *-property which led to the generalized *-property of Section IV, and to L. A. Smith and C. S. Chandersekaran for their unending tutelage in computer systems. I also wish to mention the invaluable help of Dr. D. F. Stork and A. E. Corrigan in their painstaking and incisive proofreading of the earlier versions of this report. Finally, I must thank Miss Maria Gallo for her excellent typing of this report--the job was long and difficult and she did it quickly and very well.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION I

INTRODUCTION

This report documents an extension of the research begun in "Secure Computer Systems: Mathematical Foundations"[1] and continued in "Secure Computer Systems: A Mathematical Model."[2] This extension was undertaken to investigate important facets of secure computer systems not directly covered by "A Mathematical Model." To make clear the relation between the model of the earlier volumes and the refinements of this volume, I include in this section both a brief description of the model and an outline of this report, incorporating an explanation of each refinement's place in the general scheme of the model.

The models presented in the earlier volumes of this series can be described very simply. The major elements of the models are subjects, objects, access attributes, and access rules. One can think of subjects as representing user surrogates. Similarly, objects can be thought of as representing various entities within the system including such things as data, stored programs, line printers, and teletypewriters. The access attributes in Volume II were read, execute, write, append, and control. The first four represent in a general way the mode of access suggested by their names; the last one, control, is an attribute which represents the power of a subject to give or rescind another subject's access to an object. The access rules are functions which

specify allowable changes to subjects' access to objects so that "security" is maintained. Security is defined as a particular relation between the security level of a subject and the levels of the objects it has access to at a given instant. In addition, the model's access rules prevent a certain set of circumstances wherein the potential for security compromise exists. This last property of the access rules is guaranteed by the preservation of a property called "*-property." Thus, the model describes the interrelation of subjects and objects, each with a security level, in such a way that both security (as defined) and *-property are preserved.

The next three sections of this report document three refinements to the model just described. The first refinement, found in Section II, involves the inclusion of implicit, hierarchical control. As mentioned, control was an explicit access attribute in Volume II. Viewing a directory-hierarchy machine like the Multics system as a likely vehicle for the implementation of this model, one can easily see that a more general control scheme would be very helpful. This refinement includes an implicit control scheme by distributing control throughout a hierarchically-ordered object structure, which is itself patterned after the Multics directory hierarchy.

The second refinement is included in Section III. The topic here is a concept called "current classification." The concept is included in the model to allow a vast simplification of the get-access rules of the model: a laborious check of every object currently accessed

by a subject can be replaced by a single comparison. The longer check is then included in a new rule whose purpose is to allow a subject to change its current classification; it is expected that this rule will be invoked much less frequently than the get-access rules themselves.

Section IV contains a double refinement to the *-property. It is refined to reflect the concept of current classification (of Section III) and to allow for trustworthy subjects who are exempt from *-property checks. It is emphasized that a subject may be exempted from *-property compliance only if it is demonstrated that the subject will not engage in the type of security compromise that *-property is designed to prevent.

Section V concerns a concept called "compatibility." Compatibility is a strategy for the classification of a control hierarchy which is currently required by the *-property. Section VI is a summary of the report. For the reader's convenience, three appendices are included. Appendix A is a concise list of the access rules in a standard format. Appendix B contains proofs that the rules preserve security and *-property. Appendix C contains a notational glossary: every notation, in this volume or in the two earlier ones, is listed here with a brief explanation of its meaning.

SECTION II

THE ALTERATION OF CONTROL

Introduction

An important factor in a flexible computer system is the ability to grant and rescind access privilege to users of the system. Computer systems described by "A Mathematical Model" exhibit limitations on the alteration of access privilege that are far from perfectly general. In the first place, the control attribute is explicit. Moreover, it cannot be extended during normal operation: a subject S_i has the control access attribute with respect to an object O_j

(1) if S_i created O_j, or

(2) if the control officer added that attribute to M_{ij} during abnormal operation.

In addition, a subject S_i with control over O_j can extend only those other attributes which he himself has. That is, if S_i has only the read and control access attributes relative to object O_j, then S_i cannot extend write access to subject S_u.

While modifying or lifting restrictions such as these would substantially affect the external characteristics of an implementation of the model, the control of access-privilege alteration would remain both centralized and explicit. As such, the model could not adequately describe a system such as Multics where this control is both decentralized and implicit. It is the purpose of this section to make the model more easily applicable to Multics by investigating

diffuse, implicit control over the ability to alter access privilege.

Preliminary Discussion

The type of access-privilege control we wish to incorporate into the model builds on the organization of the objects themselves. Specifically, we will deal with a set O of objects which is hierarchically organized in a directed tree structure (Figure 1). The hierarchical arrangement of objects will be dynamic, so that a means of expression which allows us to denote changes of structure easily will be desirable. The notion of a function will be used to formalize the hierarchical structure of the data: each object O will have as image the set of objects (if any) directly inferior to O. Minor changes to the current hierarchy function will then reflect alterations to the object structure itself. This framework will be developed in the next subsection.

The capability to alter access privileges will be expressed implicitly within the framework of object hierarchy. Specifically, write access to object O which is directly superior to object O_j will imply the capability to alter the access privilege of any subject to O_j (See Figure 1). The type of control thus generated is diffuse and it is implicit. In order to begin the investigation of systems with this type of control, we must start by formalizing the object-structuring function, which we shall call a hierarchy.

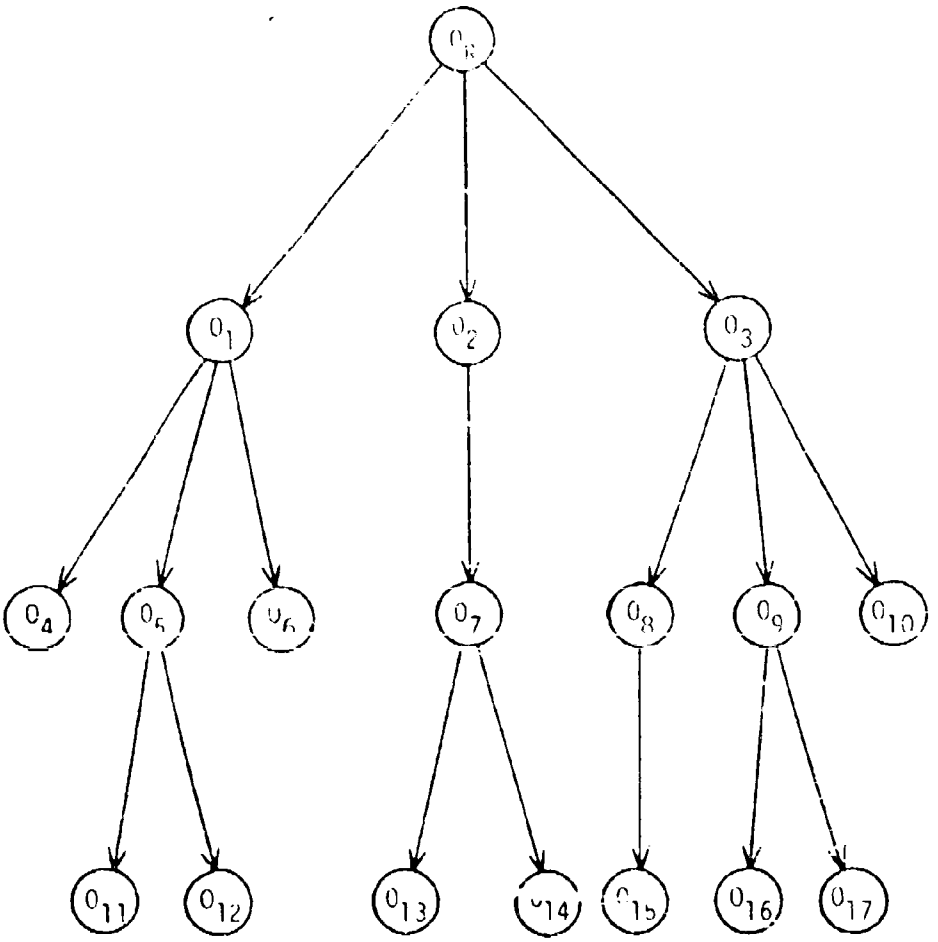


Figure 1. A Directed-Tree Structure on Objects

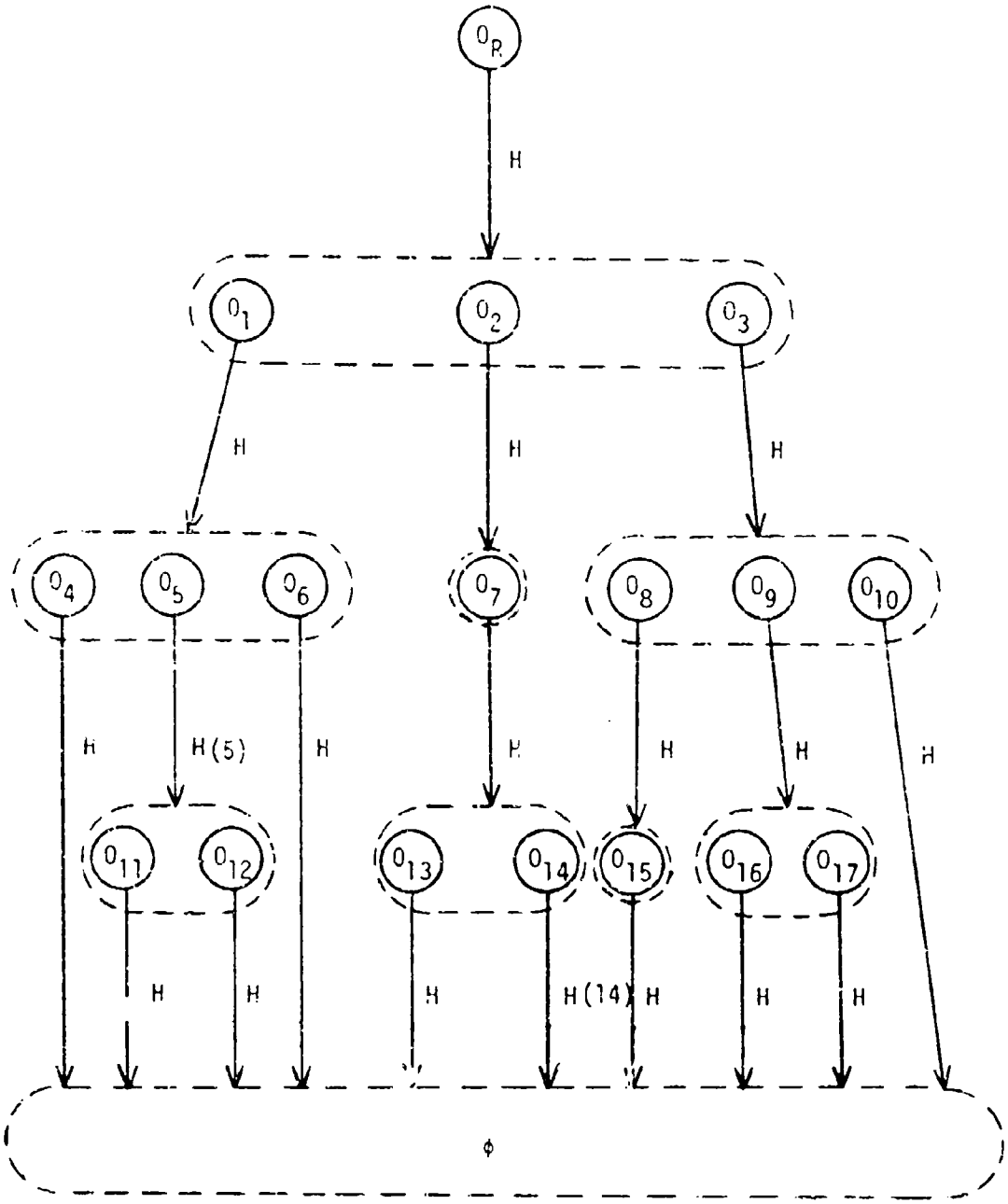


Figure 2. An Object Hierarchy H

Hierarchies

We begin by defining a set H of functions called "hierarchies."

Definition 2.1: Let $H \subseteq (P\mathcal{O})^{\mathcal{O}}$ be defined by $H \in H$ if

(1) $O_1 \neq O_2$ implies $H(O_1) \cap H(O_2) = \phi$ and

(2) there does not exist a set $\{O_1, \ldots, O_w\}$ of objects such that $O_{r+1} \in H(O_r)$ where $1 \leq r \leq w$ and $O_{w+1} = O_1$.

The interpretation of H (Figure 2) is that an object O_j is in the set $H(O)$ provided O is directly superior to O_j (or O_j is directly inferior to O). In Figure 2, $H(O_5)$ is the set of objects $\{O_{11}, O_{12}\}$ while $H(O_{14}) = \phi$ since O_{14} has no inferior objects.

Condition (1) requires therefore, that no object be directly inferior to two different objects. Condition (2) forbids the existence of a ring of objects, each directly superior to the next. In terms of graph structure, there are no directed circuits and the object structure is a tree. Notice that a hierarchy is a one-level record of connection in the object structure: more remote connections are rarely of interest in the developments to follow and are therefore suppressed in the model. That the definition of H does indeed impose a directed tree structure on $\mathcal{O}$ is established by the following proposition*.

*Propositions 2.2 and 2.3 are graph-theoretical results of a technical nature and are not vital to an understanding of the remainder of the section.

Proposition 2.2: For $H \in (P\mathcal{O})^{\mathcal{O}}$ let $E(H) = \{(O_1, O_2): O_1, O_2 \in \mathcal{O}$ and $O_2 \in H(O_1)\}$ and set $G(H)$ equal to the graph $(\mathcal{O}, E(H))$. If $H \in \mathcal{H}$, $G(H)$ is the disjoint union of directed trees and isolated points.

Proof: By (1), Indeg $O \leq 1$ for all $O \in \mathcal{O}$. By (2), O is acyclic. Thus every nontrival component of O is a directed tree and the proposition is proved.

Every $H \in \mathcal{H}$ yields a tree structure of the desired type. The converse is also true.

Proposition 2.3: Let $G = (\mathcal{O}, E)$ be the disjoint union of trees and isolated points. For every $O \in \mathcal{O}$, define $H(O) = \{O_j: (O, O_j) \in E\}$. H is a hierarchy in $\mathcal{H}$ and $G(H) \cong G$.

Proof: In a tree, Indeg $v \leq 1$ for all vertices v. The same is true for isolated vertices. Thus, (1) holds. Both trees and isolated points are acyclic. (2) holds. That $G(H) \cong G$ is immediate from the definition of H and $G(H)$.

While our definition allows there to be more than one tree, we shall consider hierarchies with a single tree to be the most relevant. Hence we shall optionally invoke property (3) which guarantees a single tree:

(3) $H^{-1}(PO - \{\phi\}) - \cup H(O) = \{O_R\}$.

The object O_R will be called the root object. If condition (3) does not hold, each object which has non-empty image under H and no inverse image will be called a root object of the system. The restriction above is mentioned only in passing since none of the results depend on it.

The set $\mathcal{H}$ of hierarchies as defined is somewhat too general for our use. Therefore, we will restrict the set of hierarchies of interest using the notion of the active set of objects. $A(M)$ is the set of object-indices which identify the objects with a nonempty column in the matrix $M \in \mathcal{M}$. Specifically, $A(M) = \{j:$ there is an $S_i \in S$ such that $M_{ij} \neq \phi\}$. In a given state, we want the active objects to be precisely the nonisolated vertices of the graph. The following definition identifies the hierarchies that satisfy this condition.

Definition 2.4: For $M \in \mathcal{M}$, set

$\mathcal{H}_M = \{H \in \mathcal{H}: H^{-1}(\phi) - \cup H(O) = \{O_j: j \notin A(M)\}\}$.

The definition of $\mathcal{H}_M$ requires $H(O_j) = \phi$ for every $j \notin A(M)$. Thus the active objects are precisely those that are in the tree portion of the structure. In a Multics setting, the active objects would be the segments in the directory structure and the terminal objects (that is, objects with no inferior objects) would represent the data segments at the "bottom" of the directory structure.

The notion of a hierarchy H is now available for use in the model of secure computer systems. In the next subsection, we shall make minor modifications to the model to incorporate the current hierarchy.

First Refinement of the Model

We shall revise the definition of a state to be a four-tuple $(b, M, f, H) \in P(S \times O \times A) \times M \times F \times H = V$ such that $H \in H_M$.

The access attributes set A is now the set $\{\underline{r}, \underline{e}, \underline{w}, \underline{a}\}$ with the same connotations as before.

Control will now be expressed implicitly. If object O_k is directly superior to $O_j \neq O_R$, then the entries M_{ij}, $1 \leq i \leq n$, are considered to reside in the object O_k directly superior to O_j. In addition, a list of the objects directly inferior to O_k, namely $H(O_k)$, is also recorded in O_k. Thus deletions or additions to access privileges to O_j can be effected by any subject having write access to O_k.

To simplify the notation somewhat, we shall partition the set of requests into five disjoint sets:

$R^{(1)}$ = requests for get- and release-access;

$R^{(2)}$ = requests for give- and rescind-access;

$R^{(3)}$ = requests for creation of objects;

$R^{(4)}$ = requests for the destruction of objects; and

$R^{(5)}$ = requests for changing classification and category set.

The intended use of the requests in $R^{(3)}$, $R^{(4)}$, and $R^{(5)}$ is obvious. Requests in $R^{(1)}$ represent initial requests for access to an object or requests to have an object removed from a subject's current-access list. Requests in $R^{(2)}$ have analogous interpretations. A give-access request represents an extending of access privilege to the named subject; a rescind-access request corresponds to revoking a subject's privilege to access a given object.

Within the model, the sets of requests are formally defined as follows:

$$R^{(1)} = RA \times S \times O \times A,\ RA = \{g, r\};$$

$$R^{(2)} = S \times RA \times S \times O \times A;$$

$$R^{(3)} = S \times O \times C \times PK \times X,\ X = \{\underline{e}, \phi\};\ \text{and}$$

$$R^{(4)} = S \times O$$

$$R^{(5)} = S \times C \times PK$$

These modifications to the model are clearly of a minor nature and they effectively include all relevant information about the current hierarchy within the current state.

Conclusion

In this section, a structure was imposed on the objects to facilitate the introduction of an implicit control attribute like that found in Multics. This change affects only those rules of Volume III which depend on the control attribute. The only such rules

involve the giving/rescinding of access or the creation/deletion of objects. New rules for these requests are included in Appendix A as rules ρ_{12}, ρ_{13}, ρ_{14}, and ρ_{15}. It is proved in Appendix B that these new rules are security-preserving and *-property-preserving, under the extended meaning of the *-property introduced in Section IV.

The set of rules $\omega_{iii} = \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_{12}, \rho_{13}, \rho_{14}, \rho_{15}\}$ defines the system $\Sigma(R, D, W(\omega_{iii}), z_0)$. By Theorems 3.2 and 3.3 of Volume II, the system is secure and satisfies *-property provided the initial state z_0 does.

SECTION III

THE INCLUSION OF CURRENT SECURITY LEVEL

Introduction

The concept of "current" security classification is directly implied by the *-property. Moreover, as was discovered in the initial attempt to use the secure computer model in the design of a security kernel,[3] not only is the use of the current security level† natural but also it can lead to dramatic simplifications of the *-property checks of rules ρ_1, ρ_2, and ρ_4 of "A Mathematical Model." This section will investigate both the justification for the current security level and the implications of its inclusion in the model.

Background

We are considering the system $\Sigma(R,D,W(\omega),z_0)$ where z_0 is secure and satisfies *-property. We begin by defining two pairs of partial functions:

$g_1(S,v) = \max \{f_2(O): (S,O,\underline{w}) \in b\}$;

$g_2(S,v) = \cup \{f_4(O): (S,O,\underline{w}) \in b\}$;

$h_1(S,v) = \max \{f_2(O): (S,O,\underline{r}) \in b\}$; and

$h_2(S,v) = \cup \{f_4(O): (S,O,\underline{r}) \in b\}$.

The domains of these functions are pairs (S,v) such that corresponding bracketed set is not empty. The intuitive interpretations of g_1 and g_2 are as follows:

$g_1(S,v)$ is the highest classification of any object O

†That is, the classification-and-category-set.[4]

currently accessed by S in the write mode in state v;

$g_2(S,v)$ is the smallest category set which contains the category set of each object O currently accessed by S in the write mode in state v.

h_1 and h_2 are interpreted similarly with "read" in place of "write".

The order imposed on the objects currently accessed by S is established by Theorem 3.1 below.

Theorem 3.1: If v satisfies *-property, then the following are true:

(1) $(S,O,\underline{w}) \in b \Rightarrow g_1(S,v) = f_2(O)$ and $g_2(S,v) = f_4(O)$;

(2) $(S,O,\underline{a}) \in b$ and $g_1(S,v)$ defined $\Rightarrow$
$g_1(S,v) \leq f_2(O)$ and $g_2(S,v) \subseteq f_4(O)$;

(3) $(S,O,\underline{r}) \in b \Rightarrow h_1(S,v) \geq f_2(O)$ and $h_2(S,v) \supseteq f_4(O)$; and

(4) $g_1(S,v)$ and $h_1(S,v)$ defined $\Rightarrow$
$g_1(S,v) \geq h_1(S,v)$ and $g_2(S,v) \supseteq h_2(S,v)$.

Proof: A direct application of the *-property.

The four conclusions above can be paraphrased as follows:

(1) If S has current write access to two different objects O_1 and O_2 in b, then O_1 and O_2 have the same classification and category set.

(2) If S has current append access to O_1 and current write access to O_2 in b, then O_1's security level

dominates O_2's security level.

(3) Conclusion (3) is a restatement of the definitions of h_1 and h_2.

(4) If S has current write access to O_1 and current read access to O_2 in b, then O_1's security level dominates the security level of O_2.

The first three of these conclusions tend to make the prospect of checking for the preservation of *-property more manageable, since the four values of g_1, g_2, h_1, and h_2 group the classifications and categories of currently-accessed objects in a natural manner. (See Figure 3.) The fourth conclusion reduces the number of important values to two. The full import of this argument is revealed in

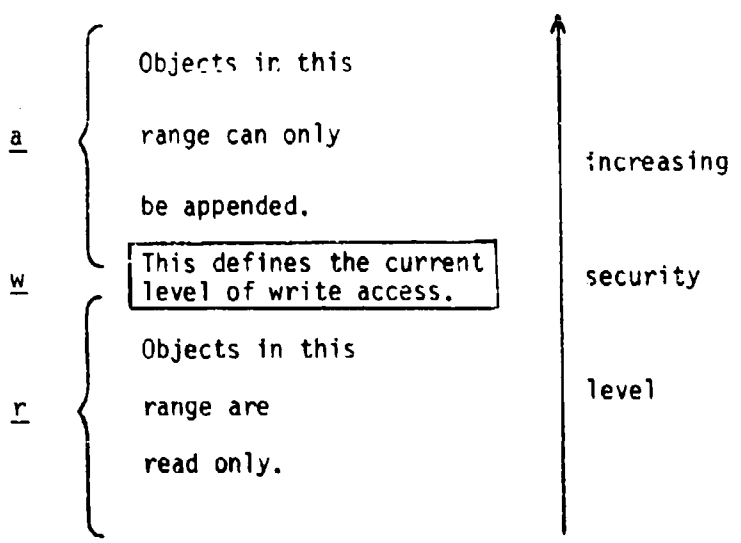


Figure 3. The Natural Ordering of Currently-Accessed Objects

Corollary 3.2.

Corollary 3.2: When $g_1(S,v)$ is defined and v satisfies the *-property, then

(i) $(S,O,\underline{a}) \in b \Rightarrow f_2(O) \geq g_1(S,v)$ and $f_4(O) \supseteq g_2(S,v)$;

(ii) $(S,O,\underline{w}) \in b \Rightarrow f_2(O) = g_1(S,v)$ and $f_4(O) = g_2(S,v)$; and

(iii) $(S,O,\underline{r}) \in b \Rightarrow f_2(O) \leq g_1(S,v)$ and $f_4(O) \subseteq g_2(S,v)$.

According to Corollary 3.2, in a state which satisfies the *-property, there is for each subject S which has current write access to some object a unique security level which equals the security level of every object currently accessed by S in the write mode. This security level simultaneously dominates those of objects being accessed in read mode and is dominated by the security

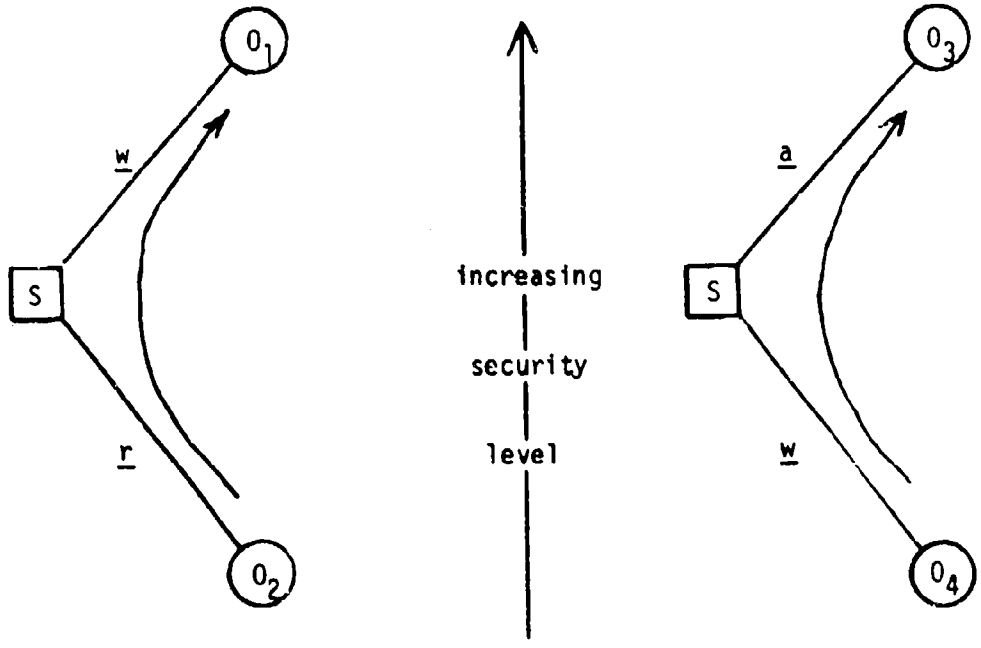


Figure 4. Information Flow and Security Level

level of every object being accessed in the append mode. (See Figure 4) The possibility of great simplification comes from the global importance of this security level for a given subject S in state v. The partial converse of Corollary 3.2 contained in Theorem 3.3 provides precisely the needed tool for simplifying the rules.

Theorem 3.3: Let $(f_5, f_6) \in C^S \times (P_K)^S$. Let $H \in H_M$ and let $v = (b,M,f,H)$ be a state such that the implications (1), (2), and (3) below hold:

(1) $(S,O,\underline{a}) \in b \Rightarrow f_2(O) \geq f_5(S)$ and $f_4(O) \supseteq f_6(S)$;

(2) $(S,O,\underline{w}) \in b \Rightarrow f_2(O) = f_5(S)$ and $f_4(O) = f_6(S)$; and

(3) $(S,O,\underline{r}) \in b \Rightarrow f_2(O) \leq f_5(S)$ and $f_4(O) \subseteq f_6(S)$.

Then v satisfies *-property.

Proof: Let $S \in S$, $O_1 \in b(S:\underline{w},\underline{a})$ and $O_2 \in b(S:\underline{r},\underline{w})$. It must be shown that $f_2(O_1) \geq f_2(O_2)$ and $f_4(O_1) \supseteq f_4(O_2)$ in order to establish that v satisfies *-property. By (1) and (2), $f_2(O_1) \geq f_5(S)$ and $f_4(O_1) \supseteq f_6(S)$. By (2) and (3), $f_5(S) \geq f_2(O_2)$ and $f_6(S) \supseteq f_4(O_2)$. The necessary relations hold by transitivity and v satisfies *-property.

Theorem 3.3 establishes that if implications (1), (2), and (3) were the definition of *-property, then checks for the preservation of *-property could be reduced in most cases to a simple comparison. The importance of this reduction in an

implementation is great enough that *-property will be redefined in Section IV to take advantage of the simpler checks. In the next subsection, the functions f_5 and f_6 of Theorem 3.3 will be added to the model for later use.

Second Refinement of the Model

We shall revise the definition of F as a subset of $C^S \times C^O \times (PK)^S \times (PK)^O \times C^S \times (PK)^S$ such that

$f = (f_1, f_2, f_3, f_4, f_5, f_6) \in F$ if and only if for all $S \in S$,

$$f_1(S) \geq f_5(S)$$

and

$$f_3(S) \supseteq f_6(S).$$

Call $f_5(S)$ the current classification of S (relative to f) and $f_6(S)$ the current category-set of S (relative to f). The current classification and current category-set will be used in the next section to redefine the *-property to take advantage of the simplification implicit in Theorem 3.3. In the remainder of this subsection, we shall discuss the simplifications which are the stimuli for the changes to come.

Define $V_{3.3}$ to be the subset of V consisting of all $v = (b,M,f,H)$ satisfying the hypotheses of Theorem 3.3. That is, $v = (b,M,f,H) \in V_{3.3}$ provided

(1) $(S,O,\underline{a}) \in b \Rightarrow f_2(O) \geq f_5(S)$ and $f_4(O) \supseteq f_6(S)$;

(2) $(S,O,\underline{w}) \in b \Rightarrow f_2(O) = f_5(S)$ and $f_4(O) = f_6(S)$; and

(3) $(S,O,\underline{r}) \in b \Rightarrow f_2(O) \geq f_5(S)$ and $f_4(O) \leq f_6(S)$.

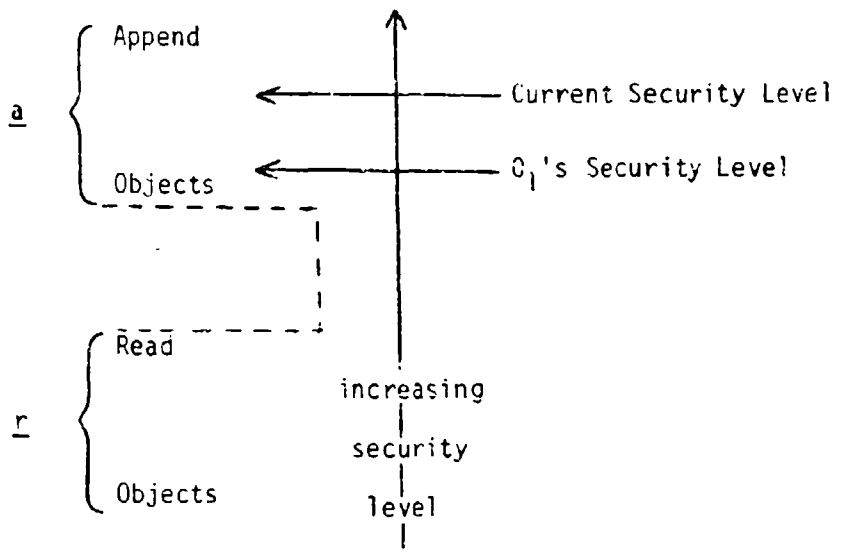
Every $v \in V_{3.3}$ satisfies *-property by Theorem 3.3. On the other hand, if v satisfies *-property and $b(S:\underline{w}) \neq \phi$ for $S \in S$, then $v \in V_{3.3}$ by Corollary 3.2. Hence, the conditions for v to be member of $V_{3.3}$ are only slightly stronger than those for v to satisfy *-property. In particular, $v = (b,M,f,H)$ can satisfy *-property and fail to be in the set $V_{3.3}$ (Figure 5) only if there is an $S \in S$ with $b(S:\underline{w}) = \phi$ and either

(1) there is an $O_1 \in b(S:\underline{a})$ with $f_2(O_1) \not\geq f_5(S)$ or $f_4(O_1) \not\leq f_6(S)$, or

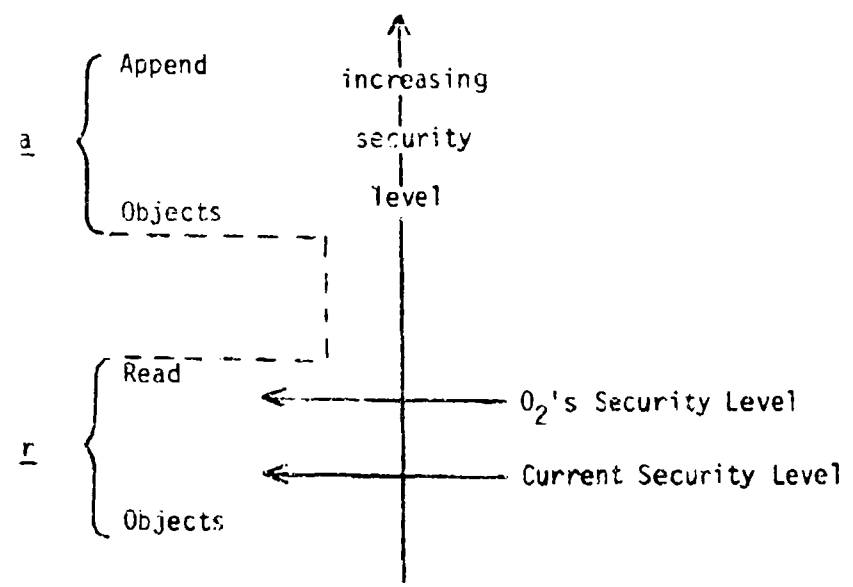
(2) there is an $O_2 \in b(S:\underline{r})$ with $f_2(O_2) \not\leq f_5(S)$ or $f_4(O_2) \not\leq f_6(S)$.

In either case, however, if *-property holds, then the security level of any object in $b(S:\underline{a})$ dominates the security level of any object in $b(S:\underline{r})$ and the exclusion of v from $V_{3.3}$ results from an incongruity between the explicit values $f_5(S)$ and $f_6(S)$ and their implicit bounds, $\inf\{f_2(O): O \in b(S:a)\}$ and $\sup\{f_2(O): O \in b(S:\underline{r})\}$ for f_5 and $\sup\{f_4(O): O \in b(S:\underline{a})\}$ and $\sup\{f_4(O): O \in b(S:\underline{r})\}$ for f_6. (These implicit bounds are represented by the lines below the $\underline{a}$ bracket and above the $\underline{r}$ bracket in Figure 5.)

To justify the elimination of this incongruity in Section IV, let us derive an alternative condition for the *-property-preserving condition $U_{\rho_1} = \phi$ of rule 1 under the assumptions that $v \in V_{3.3}$ and that S_i's security level is the infimum of the security levels of the objects in $b(S_i:\underline{w},\underline{a})$. In that situation, the following



A. Case (1): Security Level(O_1) $\not\leq$ Current Security Level(S)



B. Case (2): Security Level(O_2) $\not\leq$ Current Security Level(S)

Figure 5. State v Satisfies *-Property While v $\notin V_{3,3}$

conditions are equivalent:

$$U_{\rho_1} = \{O:\ O \in b(S_i:\underline{w},\underline{a}) \text{ and } [f_2(O_j) > f_2(O) \text{ or } f_4(O_j) \not\subseteq f_4(O)]\} = \phi$$

$$\iff [f_2(O_j) \leq f_2(O) \text{ and } f_4(O_j) \subseteq f_4(O)]$$

for all O such that $[(S_i,O,\underline{w}) \in b \text{ or } (S_i,O,\underline{a}) \in b]$

$$\iff f_2(O_j) \leq f_5(S_i) \text{ and } f_4(O_j) \subseteq f_6(S_i).$$

Clearly, then, the substitution of "$f_2(O_j) \leq f_5(S_i)$ and $f_4(O_j) \subseteq f_6(S_i)$" for "$U_{\rho_1} = \phi$" in rule ρ_1 guarantees both the fact that ρ_1 remains security-preserving and the fact that the proposition below is true:

if $v \in V_{3.3}$ and $\rho_1(R_k,v) = (D_m,v^*)$, then $v^* \in V_{3.3}$.

In fact, the same guarantee can be advanced for each of the substitutions listed below:

$[U_{\rho_1} = \phi] \iff [f_2(O_j) \leq f_5(S_i) \text{ and } f_4(O_j) \subseteq f_6(S_i)]$;

$[U_{\rho_2} = \phi] \iff [f_2(O_j) \geq f_5(S_i) \text{ and } f_4(O_j) \supseteq f_6(S_i)]$; and

$[U_{\rho_4} = \phi] \iff [f_2(O_j) = f_5(S_i) \text{ and } f_4(O_j) = f_6(S_i)]$.

It now becomes imperative that a subject be able to change his f_5 and f_6 values. This is accomplished in the model by incorporating rule ρ_{17} found in Appendix A. The checks made by ρ_{17} before granting a change of current security level are

(1) that the implications required by Theorem 3.3 vis-à-vis the objects currently accessed are satisified, and

(2) that the relations $f_1(S_i) \geq f_5(S_i)$ and $f_3(S_i) \supseteq f_6(S_i)$ hold true.

Conclusion

The rules ρ_1, ρ_2, and ρ_4 can be greatly simplified by two simple revisions. The first is a revision of *-property suggested by Theorem 3.3; this revision is found in Section IV. The second revision is the adoption of the concepts of current classification and current category set: these concepts are directly implied by the *-property itself. Little generality is lost in the rules listed in using the current security level and the new definition of *-property, since the *-property guarantees the near-equivalence of the two sets of conditions listed in the previous subsection.

SECTION IV

REVISING THE *-PROPERTY

Introduction

The *-property was introduced in "A Mathematical Model" to allow the prevention of potential compromise in secure computer systems. In this section, the *-property will be revised in two ways. The first revision was motivated in Section III and involves a new set of conditions for the definition of *-property. The second revision alters the set of subjects which are controlled by the *-property; the motivation for the second revision is contained in the next subsection.

The Background of the *-Property

The original motivation for the *-property was the potential for security compromise caused by simultaneous access of two or more objects with different security levels by a single subject. The argument for some sort of potential-compromise prevention ran as follows:

(1) a subject S with simultaneous write or append access to object O_1 and read or write access to object O_2 with security level greater than that of O_1 might put O_2 information into object O_1;

(2) if S should do so, the actual security level of the contents of O_1 would not agree with the record of O_1's security

level in the system data base;

(3) in this case, the system has lost the ability to control the situation accurately;

(4) it is not desirable for the system to lose control of the situation; hence,

(5) the system must not allow the type of simultaneous access described in (1) above.

The argument is almost syllogistic in its simplicity. The construction and use of the *-property, however, overlooked a major possibility implicit in statement (1): there may be subjects which will never mix information of different security levels as was described. The *-property was used in "A Mathematical Model" as if no subject could be trusted not to mix classified information. In this section, the *-property will be revised by removing that assumption.

The set S is the set of all subjects. Let S' represent the set of all subjects that are untrustworthy and may mix information as described. A state $v = (b,M,f,H)$ will be said to satisfy the *-property relative to S' provided that for every $S \in S'$ the *-property conditions of Theorem 3.3 are satisfied. With this definition, the the assumption that no subject can be trusted is removed and the Theorem 3.3 condition is substituted for the original *-property condition. As shall be seen later, this modification is easily integrated into the model. First, however, we shall formally alter the model in the way described.

Third Refinement of the Model

Let S' be any subset of S. A state $v = (b,M,f,H)$ with $H \in H_M$ satisfies the *-property relative to S' provided

$$S \in S' \Rightarrow \begin{cases} 0 \in b(S:\underline{a}) \Rightarrow f_2(0) \geq f_5(S) \text{ and } f_4(0) \supseteq f_6(S); \\ 0 \in b(S:\underline{w}) \Rightarrow f_2(0) = f_5(S) \text{ and } f_4(0) = f_6(S); \text{ and} \\ 0 \in b(S:\underline{r}) \Rightarrow f_2(0) \leq f_5(S) \text{ and } f_4(0) \subseteq f_6(S). \end{cases}$$

Now, v satisfies the *-property in the sense of "A Mathematical Model" provided v satisfies the *-property relative to S. Thus the new definition of *-property includes the old one but is somewhat more general.

A rule $\rho : R \times V \to D \times V$ preserves the *-property relative to S' if whenever $\rho(R_k,v) = (D_m,v^*)$ and v satisfies *-property relative to S', then v^* satisfies *-property relative to S'. Note that a proof that a rule ρ preserves *-property relative to S' can be generated from a proof that ρ preserves *-property by adding the conditions "$S_i \in S'$ " to each argument involving S_i. However, since the implications of Theorem 3.3 have been substituted for the original conditions in "A Mathematical Model," new proofs are required for the statements that the ρ_i are *-property-preserving. These proofs are included in Appendix B.

Since the change from *-property to *-property relative to S' is nearly trivial, we shall simplify discussion by using the phrase "*-property", keeping in mind that a fixed but arbitrary set S' of subjects is involved.

Conclusion

The *-property can be revised as indicated with no noticeable perturbation in the model. Moreover, with these alterations, the model can allow a vast simplification of the *-property checks as well as free a design as much as possible from excessive preventive measures. With this revision of the *-property then, the model

1. prevents untrusted subjects from degrading the system by mistake (an unexpected side-effect of a program or a bug); and
2. allows trusted subjects to operate without the extra encumbrance of the *-property.

SECTION V

CLASSIFICATION OF A CONTROL HIERARCHY

Introduction

The refinements to the model contained in Section II make it clear that objects which do not represent data are present in the model. There are, in fact, objects which represent entries in the access matrix, just as is the case for directory segments in the current Multics system. This fact causes certain problems in the design of a secure computer system. In this section, the desirability of organizing the objects in a coherent manner in order to ease these design problems is discussed. It is further shown that the proposed organization is eminently feasible.

Compatibility

The security levels of objects provides an ordering $\lhd_f$ on objects:

$$O_1 \lhd_f O_2 \iff [f_2(O_1) \leq f_2(O_2) \text{ and } f_4(O_1) \subseteq f_4(O_2)].$$

Since a hierarchy also imposes an ordering on objects, the possibility of some sort of correspondence between the two orderings presents itself as an interesting possibility.† We shall call a state v "compatible" provided the structure of $\lhd_f$ is similar to the tree-structure implied by an element of H_M. More precisely, we shall call a state $v = (b,M,f,H)$ compatible if

$$\text{for all } O \in \mathcal{O}\ [O_1 \in H(O) \Rightarrow f_2(O) \leq f_2(O_1) \ \&\ f_4(O) \subseteq f_4(O_1)].$$

†This particular object-ordering was first mentioned in (5).

That is, v is compatible provided security level is monotonically non-decreasing along any path away from the root. The next subsection will discuss the justification for requiring compatibility in a secure computer system.

Extrinsic Justification for Adopting Compatibility

The developmental work of the secure computer model has heretofore been primarily directed by current manual security procedures for classified documents. In the large, analogy was quite useful in this task: data files in an information system correspond directly to documents in a file-drawer system. Unfortunately, the analogy is not perfect. For example, the Multics analogue of the "organization" of a file drawer is the directory structure, and the directories themselves are segments, just as data-objects themselves are segments. Hence, in considering a practical secure computer system in a Multics-like environment, one is forced to consider issues beyond the purview of current security procedures. The remainder of this subsection will deal with three of those issues, leading to a justification for the adoption of compatibility.

The first issue is whether directories should be considered objects. From one point of view, directories are basically an index into the data stored in data segments. With this perspective, one would consider that the directories support the model by filling the role of a unique index to the set of objects. However, as mentioned

above, Multics directories are segments and as such are subject to alteration. Hence, the index which the directories represent is in no sense immutable. Moreover, a Multics directory contains access information to all of its inferior directories and/or data segments. Hence, some sort of control over access to directories must be enforced. Since security-related information is involved, protection for directories must be absolutely certain. The protection of objects in the model is thus of precisely the nature required for the protection of directories. Since in Multics both directories and files are segments, the inclusion of directories in the set of objects allows the protection of segments to be accomplished in a uniform manner with the secure computer model acting as a specific guide in the undertaking.

The second issue revolves around the classification of directories. It would simplify matters if nothing more than the analogues of documents (namely data segments) were required to be classified. This approach, however, is infeasible because directories are also segments, and they contain important information about inferior objects. The most obvious example is a file O whose very name is classified secret. The name of O will be part of D, so that if D is unclassified, the potential for compromise exists. Clearly, then, provision must be made to classify D appropriately to bar unauthorized users from reading the information about O which is in D. There are also many less obvious examples which can make a successful implementation

quite difficult even with classified directories.† Altogether, then, provision must be made for the classification of directories.

Suppose now that we are resolved both to include directories as objects and to allow classified directories for the reason given above. The final question is whether there is any reason to impose compatibility, thus in some sense forcing the directory structure to match the security ordering $\triangleleft_f$.

There is indeed a reason to impose compatibility. The issue here is illustrated by the situation pictured in Figure 6, a situation rendered impossible by the imposition of compatibility.

Currently access to O_2 in the Multics system is "through"

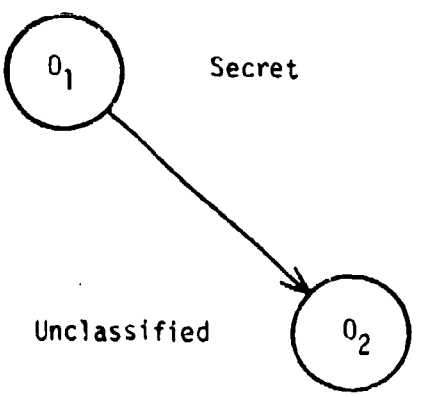


Figure 6. A Noncompatible Situation

†A full discussion of the implementation problems actually encountered in the design of a closed secure computer system can be found in Section 3.7 of the design analysis report for the Air Force Data Services Center (Reference 6).

O_1. More specifically, the name of O_2 is the path in the hierarchy from the root object O_R to O_2. Hence, a request for access to O_2 by an unclassified subject admits of two resolutions: denial of the access or provision of control mechanisms to protect O_1 while O_2 is being accessed. It is now considered unlikely that appropriate control mechanisms can be provided within the constraints of the security kernel concept.[6] Thus, it will be necessary to deny access to O_2 by unclassified subjects. Hence, no subject classified below secret could ever access objects below O_1 in the hierarchy, rendering classifications like that pictured in Figure 6 fatuous. Thus, within the constraints of practicality compatibility is a necessity. In the next subsection, it will be shown that the preservation of compatibility is not only feasible but also relatively simple.

The Preservation of Compatibility

Clearly the preservation of compatibility could be threatened only by alteration of f or H. A quick review of the rules in ω_{iii} shows that only rules ρ_{14} (create-object) and ρ_{15} (delete-object) can ever affect either f or H. Moreover, ρ_{15} does not alter f and may only disassemble part of the hierarchy tree. Thus the burden of preserving compatibility will fall on a replacement for rule ρ_{14}.

Rule ρ_{14} itself will preserve compatibility if and only if the two following conditions are satisfied:

(1) $f_2(O_j) \leq C_u$ and

(2) $f_4(O_j) \subseteq Q$.

Hence, adding these two conditions as restrictions for ρ_{14} will yield a rule which preserves compatibility.

Proposition 5.1: The rule ρ_{16} below preserves compatibility; that is, if $\rho_{16}(R_k,v) = (D_m,v^*)$ where v is compatible, then v^* is compatible. Moreover, ρ_{16} preserves security.†

$$\rho_{16}(R_k,v) = \begin{cases} (\underline{?}, v) \text{ if } R_k \notin R^{(3)}; \\ (\underline{yes},(b,M \oplus [\underline{r},\underline{w},\underline{a}]_{i,\tau(j,M)}, \alpha_1(O_j,f,C_u,Q), \beta_1(O_j,H)) \\ \quad \text{if } R_k = (S_i,O_j,C_u,Q,\phi) \in R^{(3)} \text{ and} \\ \quad [O_{s(j)} \in b(S_i:\underline{w},\underline{a})] \text{ and} \\ \quad f_2(O_j) \leq C_u \text{ and } f_4(O_j) \subseteq Q; \\ (\underline{yes},(b,M \oplus [\underline{r},\underline{e},\underline{w},\underline{a}]_{i,\tau(j,M)}, \alpha_1(O_j,f,C_u,Q),\beta_1(O_j,H)) \\ \quad \text{if } R_k = (S_i,O_j,C_u,Q,\underline{e}) \in R^{(3)} \text{ and} \\ \quad [O_{s(j)} \in b(S_i:\underline{w},\underline{a})] \text{ and} \\ \quad f_2(O_j) \leq C_u \text{ and } f_4(O_j) \subseteq Q; \\ (\underline{no},v) \text{ otherwise.} \end{cases}$$

†The function α_1 in the definition of ρ_{16} denotes the classification obtained from f by setting the new object's security level equal to (C_u,Q). β_1 is the hierarchy obtained from H by adding a new object directly below O_j. Both α_1 and β_1 are defined in Appendix C.

Proof: Write $v = (b,M,f,H)$ and $v^* = (b^*,M^*,f^*,H^*)$. Note that f and f^* differ only on $O_{\tau(j,M)}$. Pick $O \neq O_j$. Then $H^*(O) = H(O)$. Hence since v is compatible, $f_t(O)$ is dominated by $f_t(O_1)$ for $O_1 \in H(O)$ and $t = 2, 4$. Consider O_j. $H^*(O_j)$ equals either $H(O_j)$ or $H(O_j) \cup \{O_{\tau(j,M)}\}$, and the check for compatibility reduces to a check vis-à-vis O_j and $O_{\tau(j,M)}$ where $R_k \in R^{(3)}$; $O_{s(j)} \in b(S_i:\underline{w},\underline{a})$; $f_2(O_j) \leq C_u$; and $f_4(O_j) \subseteq Q$. But $f_2^*(O_{\tau(j,M)}) = C_u$ and $f_4^*(O_{\tau(j,M)}) = Q$ so that v^* is compatible and ρ_{16} is compatibility-preserving as claimed.

Since ρ_{16} is a refinement of ρ_{14} which is security-preserving, ρ_{16} is itself security-preserving.

If one desires a secure computer system which exhibits only compatible states satisfying the *-property, one can use the set of rules

$$\omega_{iii}' = \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_{12}, \rho_{13}, \rho_{15}, \rho_{16}, \rho_{17}\}$$

together with a compatible, secure initial state z_0 which satisfies the *-property.

Conclusion

Compatibility is currently required by practical considerations. In addition, it can be provided by adding one further condition to the create-object rule. Thus, compatibility is both a desirable and a feasible refinement to the secure computer model.

SECTION VI

SUMMARY

In this report, the secure computer model is extended in several ways.

The first extension allows a hierarchical ordering of the objects and for implicit control over the objects. This revision clearly incorporates certain control structures into the set of objects. With appropriate interpretation of the access attributes as applied to control objects in the hierarchy, the model can now be applied to a Multics-like information system in a very direct way.

The second extension is the introduction of current classification and current category set (f_5 and f_6, respectively). This revision made possible a vast simplification of the *-property check in the various rules.

The third extension alters the *-property to allow for trustworthy subjects. This revision makes the development of the *-property sounder and it gives the model more flexibility.

Finally, a number of practical considerations are discussed, leading to the conclusion that enforcing compatibility, a discipline of nondecreasing security level on the object hierarchy, is required by the *-property in a Multics-like environment. The inclusion of this discipline is achieved by the replacement of one rule in ω_{iii} by another very similar one, resulting in a secure computer system

which preserves both the extended *-property and compatibility.

DEB:mg

APPENDIX A

THE RULES

In this appendix, the rules of this volume are listed in numerical order. The set ω_{iii} of Section II comprises rules 1, 2, 3, 4, 5, 12, 13, 14, 15, and 17. The set ω'_{iii} of Section V, which guarantees the preservation of compatibility, consists of the rules 1, 2, 3, 4, 5, 12, 13, 15, 16, and 17. Rules 1 - 5 are rules retained from "A Mathematical Model." Rules 12 - 17 are new rules. The proof of rule 16 appears in Section V as Proposition 5.1 (page 33); the remaining proofs are in Appendix B.

There is a standard format for the presentation of rules in this appendix. The domain of ρ_i is a description of the form a request that rule ρ_i handles will take. The semantics of ρ_i is a brief explanation of the situation that rule ρ_i is designed to arbitrate. The *-property function σ_i is a Boolean function which specifies the conditions which must be satisfied before a positive decision is allowed for an untrustworthy subject--that is, a subject in S'. If the *-property does not affect the rule, the *-property function is listed as a tautology (always TRUE) and σ_i does not appear in the rule itself. Next the rule is presented in an abbreviated functional form. Finally, an algorithm for ρ_i is presented. Any unfamiliar notation can be found listed in the Notational Glossary in Appendix C.

Rule 1: get-read

Domain of ρ_1: all $R_k = (g, S_i, O_j, \underline{r}) \in R^{(1)}$.

Semantic: Subject S_i requests access to object O_j in read ($\underline{r}$) mode.

*-property function:

$$\sigma_1(R_k, v) = \text{TRUE} \iff [f_5(S_i) \geq f_2(O_j) \;\&\; f_6(S_i) \supseteq f_4(O_j)].$$

The rule:

$$\rho_1(R_k, v) = \begin{cases} (\underline{?}, v) & \text{if } R_k \notin \text{domain of } \rho_1; \\ (\text{yes}, \text{augb}(R_k, v)) & \text{if } [R_k \in \text{domain of } \rho_1] \\ & \;\&\; [\underline{r} \in M_{ij}] \\ & \;\&\; [f_1(S_i) \geq f_2(O_j)] \\ & \;\&\; [f_3(S_i) \supseteq f_4(O_j)] \\ & \;\&\; [S_i \notin S' \text{ or } \sigma_1(R_k, v)]; \\ (\underline{\text{no}}, v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_1:

```
if R_k ∉ domain of ρ1 then ρ1(R_k,v) = (?,v);
    else if r ∈ M_ij
        and <[S_i ∈ S' and σ1(R_k,v)]
            or [S_i ∉ S' and f1(S_i) ≥ f2(O_j) and f3(S_i) ⊇ f4(O_j)]>
        then ρ1(R_k,v) = (yes, augb(R_k,v));
            else ρ1(R_k,v) = (no, v);
end;
```

Rule 2: get-append

Domain of ρ_2: all $R_k = (g, S_i, O_j, \underline{a}) \in R^{(1)}$.

Semantics: Subject S_i requests access to object O_j in append ($\underline{a}$) mode.

*-property function:

$$\sigma_2(R_k, v) = \text{TRUE} \iff [f_5(S_i) \leq f_2(O_j) \;\&\; f_6(S_i) \subseteq f_4(O_j)].$$

The rule:

$$\rho_2(R_k, v) = \begin{cases} (\underline{?}, v) & \text{if } R_k \notin \text{domain of } \rho_2; \\ (\underline{yes}, \text{augb}(R_k, v)) & \text{if } [R_k \in \text{domain of } \rho_2] \\ & \;\&\; [\underline{a} \in M_{ij}] \\ & \;\&\; [S_i \notin S' \text{ or } \sigma_2(R_k, v)]; \\ (\underline{no}, v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_2:

```
if  R_k ∉ domain of ρ2 then ρ2(R_k,v) = (?,v);
   else if a ∈ M_ij
     and <[S_i ∈ S' and σ2(R_k,v)] or [S_i ∉ S']>
    then ρ2(R_k,v) = (yes,augb(R_k,v));
         else ρ2(R_k,v) = (no,v);
end;
```

Rule 3: get-execute

Domain of ρ_3: all $R_k = (g, S_i, O_j, \underline{e}) \in R^{(1)}$.

Semantics: Subject S_i requests access to object O_j in execute ($\underline{e}$) mode.

*-property function:

$$\sigma_3(R_k, v) = \text{TRUE}.$$

The rule:

$$\rho_3(R_k, v) = \begin{cases} (\underline{?}, v) & \text{if } R_k \notin \text{domain of } \rho_3; \\ (\underline{yes}, \text{augb}(R_k, v)) & \text{if } [R_k \in \text{domain of } \rho_3] \\ & \quad \& \; [\underline{e} \in M_{ij}]; \\ (\underline{no}, v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_3:

```
if Rk ∉ domain of ρ3 then  ρ3(Rk,v) = (?,v);
  else if e ∈ Mij  then ρ3(Rk,v) = (yes,augb(Rk,v));
     else  ρ3(Rk,v) = (no,v);
end;
```

Rule 4: get-write

Domain of ρ_4: all $R_k = (g, S_i, O_j, \underline{w}) \in R^{(1)}$.

*-property function:

$$\sigma_4(R_k, v) = \text{TRUE} \iff [f_5(S_i) = f_2(O_j) \;\&\; f_6(S_i) = f_4(O_j)].$$

The rule:

$$\rho_4(R_k, v) = \begin{cases} (\underline{?}, v) & \text{if } R_k \notin \text{domain of } \rho_{14}; \\ (\underline{yes}, \text{augb}(R_k, v)) & \text{if } [R_k \in \text{domain of } \rho_4] \\ & \;\&\; [\underline{w} \in M_{ij}] \\ & \;\&\; [f_1(S_i) \geq f_2(O_j)] \\ & \;\&\; [f_3(S_i) \supseteq f_4(O_j)] \\ & \;\&\; [S_i \notin S' \text{ or } \sigma_4(R_k, v)]; \\ (\underline{no}, v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_4:

```
if R_k ∉ domain of ρ_4 then ρ_4(R_k,v) = (?,v);
  else if w ∈ M_ij
    and <[S_i ∉ S' and f_1(S_i) ≥ f_2(O_j) and f_3(S_i) ⊇ f_4(O_j)]
         or [S_i ∈ S' and σ_4(R_k,v)]>
    then ρ_4(R_k,v) = (yes,augb(R_k,v));
         else ρ_4(R_k,v) = (no,v);
end;
```

Rule 5: release-read/execute/write/append

Domain of ρ_5: all $R_i = (r, S_i, O_j, \underline{x}) \in R^{(1)}$.

Semantics: Subject S_i signals the release of access to object O_j in mode $\underline{x}$, where $\underline{x}$ is $\underline{r}$ (read), $\underline{e}$ (execute), $\underline{w}$ (write), or $\underline{a}$ (append).

*-property function:

$$\rho_5(R_k, v) = \text{TRUE}.$$

The rule:

$$\rho_5(R_k, v) = \begin{cases} (\underline{yes}, \text{dimb}(R_k, v)) & \text{if } R_k \in \text{domain of } \rho_5; \\ (\underline{?}, v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_5:

```
if Rk ∉ domain of ρ5 then ρ5(Rk,v) = (?,v);
   else ρ5(Rk,v) = (yes, dimb(Rk,v));
end;
```

Rule 12: give-read/execute/write/append

Domain of ρ_{12}: all $R_k = (S_\lambda, g, S_i, O_j, \underline{x}) \in R^{(2)}$ with $O_j \neq O_R$.

Semantics: Subject S_λ gives subject S_i the right of access to object O_j in mode $\underline{x}$ where $\underline{x}$ is $\underline{r}$, $\underline{e}$, $\underline{w}$, or $\underline{a}$.

*-property function:

$$\sigma_{12}(R_k, v) = \text{TRUE}.$$

The rule:

$$\rho_{12}(R_k, v) = \begin{cases} (\underline{?}, v) & \text{if } R_k \notin \text{domain of } \rho_{12}; \\ (\underline{yes}, (b, M \oplus \lceil \underline{x} \rceil_{ij}, f, H)) & \text{if } [R_k \in \text{domain of } \rho_{12}] \\ & \quad \& \; [O_{s(j)} \in b(S_\lambda : \underline{w})]; \\ (\underline{no}, v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_{12}:

```
if R_k ∉ domain of ρ12 then ρ12(R_k,v) = (?,v);
  else if O_s(j) ∈ b(S_λ:w)
     then ρ12(R_k,v) = (yes, (b,M ⊕ [x]_ij,f,H));
       else ρ12(R_k,v) = (no,v);
end;
```

Rule 13: rescind-read/execute/write/append

Domain of ρ_{13}: all $R_k = (S_\lambda, r, S_i, O_j, \underline{x}) \in R^{(2)}$ with $O_j \neq O_R$.

Semantics: Subject S_λ rescinds S_i's privilege of access to object O_j in mode $\underline{x}$ when $\underline{x}$ is $\underline{r}$, $\underline{e}$, $\underline{w}$, or $\underline{a}$.

*-property functions:

$$\sigma_{13}(R_k, v) = \text{TRUE}.$$

The rule:

$$\rho_{13}(R_k, v) = \begin{cases} (\underline{?}, v) & \text{if } R_k \notin \text{domain of } \rho_{13}; \\ (\underline{yes}, (b - \{(S_i, O_j, x)\}, M \ominus [\underline{x}]_{ij}, f, H)) & \text{if } [R_k \in \text{domain of } \rho_{13}] \\ & \quad \& [O_{s(j)} \in b(S_\lambda : \underline{w})]; \\ (\underline{no}, v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_{13}:

```
if R_k ∉ domain of ρ13 then ρ13(R_k,v) = (?,v);
   else if O_s(j) ∈ b(S_λ:w) then
                ρ13(R_k,v) = (yes, (b - {(S_i,O_j,x)},M ⊖ [x]_ij,f,H))
           else ρ13(R_k,v) = (no,v);
end;
```

Rule 14: create-object

Domain of ρ_{14}: all $R_k = (S_i, O_j, C_u, Q, x) \in R^{(3)}$ with $O_j \neq O_R$.

Semantics: Subject S_i requests the "creation" (i.e., attachment) of an object directly below object O_j. The new object is to have classification C_u and category set Q. If $x = \underline{e}$, S_i wishes to be given $\underline{r}$, $\underline{e}$, $\underline{w}$, and $\underline{a}$ access to the new object; if $x = \phi$, S_i wishes only $\underline{r}$, $\underline{w}$, and $\underline{a}$ access.

*-property function:

$$\sigma_{14}(R_k, v) = \text{TRUE}.$$

The rule:

$$\rho_{14}(R_k, v) = \begin{cases} (\underline{?}, v) & \text{if } R_k \notin \text{domain of } \rho_{14}; \\ (\underline{yes}, (b, M \oplus [\underline{r}, \underline{w}, \underline{a}]_{i,\tau(j,M)}, \alpha_1(O_{\tau(j,M)}, f, C_u, Q), \beta_1(O_j, H, M)) & \\ & \text{if } [R_k = (S_i, O_j, C_u, Q, \phi) \in R^{(3)}] \\ & \& [O_j \in b(S_i : \underline{w}, \underline{a})]; \\ (\underline{yes}, (b, M \oplus [\underline{r}, \underline{e}, \underline{w}, \underline{a}]_{i,\tau(j,M)}, \alpha_1(O_{\tau(j,M)}, f, C_u, Q), \beta_1(O_j, H, M)) & \\ & \text{if } [R_k = (S_i, O_j, C_u, Q, \underline{e}) \in R^{(3)}] \\ & \& [O_j \in b(S_i : \underline{w}, \underline{a})]; \\ (\underline{no}, v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_{14}:

if $R_k \notin R^{(3)}$, the domain of ρ_{14}, then $\rho_{14}(R_k,v) = (?,v)$;

else if $O_j \quad \varepsilon\ b(S_i:w,a)$ then

do;

$\Phi = \{r,w,a\}$;

if $\chi = e$ then $\Phi = \Phi \cup \{e\}$;

$\rho_{14}(R_k,v) = (yes,(b,M, \circledast [\Phi]_{i,\tau(j,M)},\alpha_1(O_{\tau(j,M)},f,C_u,Q),\beta_1(O_j,H,M))$;

end;

else $\rho_{14}(R_k,v) = (no,v)$;

end;

Rule 15: delete-object

Domain of ρ_{15}: all $R_k = (S_i, O_j) \in R^{(4)}$ with $O_j \neq O_R$.

Semantics: Subject S_i requests that object O_j and all objects inferior to O_j be "deleted" (i.e., detached from the hierarchy).

*-property function:

$$\sigma_{15}(R_k, v) = \text{TRUE}.$$

The rule:

$$\rho_{15}(R_k, v) = \begin{cases} (\underline{?}, v) & \text{if } R_k \notin \text{domain of } \rho_{15}; \\ (\underline{yes}, (\Delta_1(j,b), \Delta_2(j,M), f, \beta_2(O_j,H)) & \\ & \text{if } [R_k \in \text{domain of } \rho_{15}] \\ & \& \ [O_{s(j)} \in b(S_i : \underline{w})]; \\ (\underline{no}, v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_{15}:

if $R_k \in R^{(4)}$, the domain of ρ_{15}, then $\rho_{15}(R_k, v) = (\underline{?}, v)$;
  else if $O_{s(j)} \in b(S_i : \underline{w})$ then
    $\rho_{15}(R_k, v) = (\underline{yes}, (\Delta_1(j,b), \Delta_2(j,M), f, \beta_2(O_j,H))$;
  else $\rho_{15}(R_k, v) = (\underline{no}, v)$;
end;

Rule 16: create-object(preserving compatibility)

Domain of ρ_{16}: all $R_k = (S_i, O_j, C_u, Q, x) \in R^{(3)}$ with $O_j \neq O_R$.

Semantics: Subject S_i requests the "creation" (i.e., attachment) of an object directly below object O_j. The new object is to have classification C_u and category set Q. S wishes to receive access attributes r w, and a to the new object. S_i also wishes e access if x = e.

*-property function:

$$\sigma_{16}(R_k, v) = \text{TRUE}.$$

The rule:

$$\rho_{16}(R_k, v) = \begin{cases} (?, v) & \text{if } R_k \notin \text{domain of } \rho_{16}; \\ (\underline{yes}, (b, M \oplus [\underline{r}, \underline{w}, \underline{a}]_{i,\tau(j,M)}, \alpha_1(O_{\tau(j,M)}, f, C_u, Q), \beta_1(O_j, H, M)) & \\ \quad \text{if } [R_k = (S_i, O_j, C_u, Q, \phi) \in R^{(3)}] & \\ \quad \& \; [O_j \in b(S_i : \underline{w}, \underline{a})] & \\ \quad \& \; [f_2(O_j) \leq C_u \text{ and } f_4(O_j) \subseteq Q]; & \\ (\underline{yes}, (b, M \oplus [\underline{r}, \underline{e}, \underline{w}, \underline{a}]_{i,\tau(j,M)}, \alpha_1(O_{\tau(j,M)}, f, C_u, Q), \beta_1(O_j, H, M)) & \\ \quad \text{if } [R_k = (S_i, O_j, C_u, Q, \underline{e}) \in R^{(3)}] & \\ \quad \& \; [O_j \in b(S_i : \underline{w}, \underline{a})] & \\ \quad \& \; [f_2(O_j) \leq C_u \text{ and } f_4(O_j) \subseteq Q]; & \\ (\underline{no}, v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_{16}:

<u>if</u> $R_k \notin R^{(3)}$, the domain of ρ_{16}, <u>then</u> $\rho_{16}(R_k,v) = (\underline{?},v)$;

<u>else if</u> $O_j \in b(S_i:\underline{w},\underline{a})$ <u>then</u>

<u>do</u>;

$\Phi = \{\underline{r},\underline{w},\underline{a}\}$;

<u>if</u> $\chi = \underline{e}$ <u>then</u> $\Phi = \Phi \cup \{\underline{e}\}$;

$\rho_{16}(R_k,v) = (yes,(b,M, \Theta\,[\Phi]_{i,\tau(j,M)},\alpha_i(O_{\tau(j,M)},f,C_u,Q),\beta_1(O_j,H,M))$;

<u>end</u>;

<u>else</u> $\rho_{16}(R_k,v) = (\underline{no},v)$;

<u>end</u>;

Rule 17: change-security-level

Domain of ρ_{17}: all $R_k = (S_i, C_u, Q) \in R^{(5)}$.

Semantics: Subject S_i requests a change in his current security level (that is, his f_5 and f_6 values) to C_u and Q.

*-property function:

$$\begin{aligned}\rho_{17}(R_k,v) = \text{TRUE} \iff & [f_1(S_i) \geq C_u \text{ and } f_3(S_i) \supseteq Q] \\ & \& [0 \in b(S_i:\underline{a}) \Rightarrow C_u \leq f_2(0) \text{ and } Q \subseteq f_4(0)] \\ & \& [0 \in b(S_i:\underline{w}) \Rightarrow C_u = f_2(0) \text{ and } Q = f_4(0)] \\ & \& [0 \in b(S_i:\underline{r}) \Rightarrow C_u \geq f_2(0) \text{ and } Q \supseteq f_4(0)].\end{aligned}$$

The rule:

$$\rho_{17}(R_k,v) = \begin{cases} (\underline{?},v) & \text{if } R_k \notin \text{domain of } \rho_{17}; \\ (\underline{yes},(b,M,\alpha_2(f,S_i,C_u,Q),H)) & \text{if } [R_k \in \text{domain of } \rho_{17}] \\ & \quad \& [S_i \notin S' \text{ or } \sigma_{17}(R_k,v)]; \\ (\underline{no},v) & \text{otherwise.} \end{cases}$$

Algorithm for ρ_{17}:

```
if R_k ∉ domain of ρ17 then ρ17(R_k,v) = (?,v);
  else if [S_i ∉ S' or σ17(R_k,v)] then ρ17(R_k,v) = (yes,(b,M,α2(f,S_i,C_u,Q),H));
    else ρ17(R_k,v) = (no,v);
end;
```

APPENDIX B

PROOFS OF THE RULES

In this appendix are gathered all new proofs for the various rules of Volume III. The proofs of the rules contained in Volume II are not included here. Since the proof that rule ρ_{16} is security-preserving is contained in Proposition 5.1, it is omitted here.

For rule 12, the full justification of the correspondence between the functional specification of the rule and its algorithm is given in full to indicate the form such a verification takes. Similar demonstrations for the other rules are omitted.

Proposition B.0: A rule ρ preserves *-property relative to S' if the following implication is valid:

if $v = (b,M,f,H)$ satisfies *-property relative to S'; $\rho(R_k,v) = (D_m,v^*)$; $v^* = (b^*,M^*,f,H^*)$; $H^* \in H_M$; and $(S_i,O_j,\underline{x}) \in b^* - b$, then $S \notin S'$ or O_j satisfies the first, second, or third *-property condition as $\underline{x}$ is $\underline{a}$, $\underline{w}$, or $\underline{r}$.

Proof: Assume the implication is valid. Let $S \in S'$ and suppose $O \in b^*(S: \underline{a},\underline{w},\underline{r})$.

If $O \in b^*(S: \underline{a})$, then either $O \in b(S:\underline{a})$ or $(S,O,\underline{a}) \in b^* - b$. In either case, the first *-property condition holds, either by the assumption on v or by the implication.

Similarly, if $0 \in b^*(S: \underline{w})$ (respectively, $b^*(S:\underline{r})$), then either $0 \in b(S: \underline{w})$ (respectively, $b(S: \underline{r})$) or $(S,0,\underline{w}) \in b^* - b$ (respectively, $(S,0,\underline{r}) \in b^* - b$). Again in either case, the second (respectively, third) *-property condition holds, either by the assumption on v or by the implication.

Since the definition of *-property is satisified, v^* satisfies *-property as claimed.

Proposition B.1: Rules 1, 2, and 4 preserve *-property relative to S'.

Proof: Let $u = 1$, 2, or 4 and $\underline{x}_1 = \underline{a}$, $\underline{x}_2 = \underline{w}$, and $\underline{x}_4 = \underline{r}$. Suppose $v = (b,M,f,H)$ satisfies *-property relative to S'; $\rho_u(R_k,v) = (D_m, v^*)$; and $v^* = (b^*,M^*,f^*,H^*)$. By Proposition B.0, it suffices to show that

(1) $f^* = f$;

(2) $H^* \in H_M$; and

(3) $(S_i,O_j,\underline{x}) \in b^* - b$ implies

$S_i \notin S'$ or O_j satisfies the appropriate *-property condition.

But by ρ_u, $v^* = v$ or $aug(R_k,v)$ so that in either case $f^* = f$ and $H^* = H \in H_M$. Now $b^* \neq b$ iff $v^* = aug(R_k,v)$. Hence $b^* \neq b$ implies $b^* - b = \{(S_i,O_j,\underline{x}_u)\}$ where $R_k = (g, S_i,O_j,\underline{x}_u)$. If $S_i \in S'$, then since $v^* = aug(R_k,v)$, $\sigma_u(R_k,v) = TRUE$, which is by definition equivalent to the appropriate *-property condition.

Thus by Proposition B.0, v* satisfies *-property relative to S'.

Proposition B.2: Rules 3, 5, 12, 13, 14, 15, and 16 preserve *-property relative to S'.

Proof: This proposition follows directly from Proposition B.0. For rules 5, 12, 13, 14, 15, and 16, the premise $(S_i,O_j,\underline{x}) \in b^* - b$ of the implication above is never true so that the implication itself is trivially true. Hence the listed rules do indeed preserve *-property relative to S'. For rule 3, $b^* - b \neq \phi$ implies $b^* - b = \{(S_i,O_j,\underline{e})\}$ so that the conclusion of the implication is vacuously true. Hence rule 3 also preserves the *-property relative to S'.

Proposition B.3: Rule 12 is security-preserving.

Proof: If $v = (b,M,f,H)$, $v^* = (b^*,M^*,f^*,H^*)$ and $\rho_{12}(R_k,v) = (D_m,v)$, then $b^* = b$ and $f^* = f$. If v is secure, every $(S_i,O_j,\underline{x})$ in $b = b^*$ satisfies SC rel f (hence SC rel f*). Thus v* must be secure.

Proposition B.4: The listed algorithm calculates ρ_{12}.

Proof:

(i) Let $R_k \notin R^{(2)}$. The condition on line 1 of the algorithm is satisfied, so that $\rho_{12}(R_k,v)$ is set equal to $(\underline{?},v)$ as desired and the algorithm terminates in this case.

(ii) Let $R_k = (S_\lambda,r,S_i,O_j,\underline{x}) \in R^{(2)}$. The condition of line 1

is not satisfied so line 2 is used. The condition of line 2 is satisfied so that $\rho_{12}(R_k,v)$ is set equal to $(\underline{?},v)$ as desired; the algorithm terminates.

(iii) Let $R_k = (S_\lambda,g,S_i,O_j,\underline{x}) \in R^{(2)}$ and let $\underline{w} \in M_{\lambda,s(j)}$. The conditions on lines 1 and 2 are not satisfied; line 3 is used. The condition of line 3 is satisfied and $\rho_{12}(R_k,v)$ is set equal to $(\underline{yes}, (b, M \circledast [\underline{x}]_{ij}, f, H))$ as desired. The algorithm terminates.

(iv) Suppose none of the conditions of (i), (ii), or (iii) hold. The condition of line 1 cannot be satisfied since (i) doesn't hold. The condition of line 2 cannot hold since (ii) doesn't hold. Since (iii) doesn't hold, the condition on line 3 is not satisfied so that line 4 of the algorithm is invoked, setting $\rho_{12}(R_k,v)$ equal to $(\underline{no}, v)$ as desired. The algorithm terminates.

Proposition B.5: Rule 13 is security-preserving.

Proof: Let $v = (b,M,f,H)$, $v^* = (b^*,M^*,f^*,H^*)$, and $\rho_{13}(R_k,v) = (D_m,v^*)$. By rule 13, $b^* \subseteq b$ and $f^* = f$. Hence if v is secure, v^* is secure.

Proposition B.6: Rule 14 is security-preserving.

Proof: If $v = (b,M,f,H)$, $v^* = (b^*,M^*,f^*,H^*)$, and $\rho_{14}(R_k,v) = (D_m,v^*)$, then $b^* = b$ and $f^* = \alpha_1(O_j,f,C_u,Q)$ provided

$R_k = (S_i, O_j, C_u, Q, \underline{x}) \in R^{(3)}$ and $f^* = f$ otherwise. Since f^* and f agree on subjects and on active objects, v^* is secure provided v is secure.

Proposition B.7: Rule 15 is security-preserving.

Proof: If $v = (b, M, f, H)$, $v^* = (b^*, M^*, f^*, H^*)$, and $\rho_{15}(R_k, v) = (D_m, v^*)$, then $b^* \subseteq b$ and $f^* = f$. Thus v^* is secure provided v is.

Proposition B.8: Rule 16 is security-preserving.

Proof: See Proposition 5.1, page 33.

Proposition B.9: Rule 17 is security-preserving, and preserves the *-property relative to S'.

Proof: If $v = (b, M, f, H)$, $v^* = (b^*, M^*, f^*, H^*)$, and $\rho_{17}(R_k, v) = (D_m, v^*)$, then $b^* = b$ and $f^* = f$ or $a_2(f, S_i, C_u, Q)$. In either case, neither b nor the values of f_1, f_2, f_3, or f_4 have changed so that security is preserved.

If $f^* = f$ or if $S_i \notin S'$, then it is immediate that ρ_{17} preserves *-property relative to S' by Proposition B.0. If $f^* \neq f$ and $S_i \in S'$, then $f^* = a_2(f, S_i, C_u, Q) \in R_k \in$ domain of ρ_{17}, and $\sigma_{17}(R_k, v)$ is true. Clearly $f^* \in F$ since $f_5^*(S_i) \leq f_1^*(S_i) = f_1(S_i)$ and $f_6^*(S_i) = Q \subseteq f_3^*(S_i) = f_3(S_i)$ by the first condition of ρ_{17}.

Moreover,

$$0 \in b^*(S_i:\underline{a}) = b(S_i:\underline{a}) \Rightarrow [f_2^*(0) = f_2(0) \geq C_u = f_5^*(S_i) \;\&\; f_4^*(0) = f_4(0) \supseteq Q = f_6^*(S_i)];$$

$$0 \in b^*(S_i:\underline{w}) = b(S_i:\underline{w}) \Rightarrow [f_2^*(0) = f_2(0) = C_u = f_5^*(S_i) \;\&\; f_4^*(0) = f_4(0) = Q = f_6^*(S_i)]; \text{ and}$$

$$0 \in b^*(S_i:\underline{r}) = b(S_i:\underline{R}) \Rightarrow [f_2^*(0) = f_2(0) \leq C_u = f_5^*(S_i) \;\&\; f_4^*(0) = f_4(0) \subseteq Q = f_6^*(S_i)].$$

Thus if v satisfies *-property relative to S', then v* satisfies *-property relative to S'. Hence ρ_{17} also preserves the *-property relative to S' and the assertion is proved.

APPENDIX C

NOTATIONAL GLOSSARY

In this appendix are listed the major notations used in the Secure Computer Systems series. There are three lists--a Roman alphabet list, a Greek alphabet list, and a symbol list. Each entry has a brief description of the concept involved and a reference to the principal appearances of the notation in the three volumes. A reference is in the form $(n_1; n_2; n_3)$ where n_1 is a page number in Volume I, n_2 is a page number in Volume II, and n_3 is a page number in Volume III.

Roman Alphabet List

a, append	the alter-only attribute in the set of access attributes. (-; 22; 11)
A	the set of access attributes. (-; 22;11)
A(M)	the set of active object indices; $\{j: 1 \leq j \leq m \text{ and } M_{ij} \neq \phi \text{ for some } i\}$. (-; 39; -)
augb(R_k,v)	denotes the addition of the triple specified by R_k to b; if $R_k = (g,S_i,O_j,\underline{x})$ and v = (b,M,f,H), then augb(R_k,v) = $(b \cup \{(S_iO_j,\underline{x})\},M,f,H)$. (-; 38; -)

b, b*	a record of current access; a subset of $P(S \times O \times A)$. (17;25;-)
b(S:$\underline{x}$,$\underline{y}$,...,$\underline{z}$)	the set of objects O accessed by S in mode $\underline{x}$ or $\underline{y}$ or . . . or $\underline{z}$, according to b; {O; O ϵ $\mathcal{O}$ and [(S,O,$\underline{x}$) ϵ b or . . . or (S,O,$\underline{z}$) ϵ b]}. (-;27;-)
c	an element of RA denoting "create." (-;22;-)
$\underline{c}$, $\underline{control}$	the control attribute in the set A in Volume II. (-;22;-)
C	the set of classifications. (14;22;-)
C_u	an arbitrary classification from the set C. (14;22;-)
d	an element of RA denoting "delete." (-;22;-)
D	the set of decisions. (15;23;-)
D_m	an arbitrary decision from the set D. (15;23;-)
dimb(R_k,v)	denotes the deletion of the triple specified by R_k from b; if $R_k = (r,S_i,O_j,\underline{x})$ and $v = (b,M,f,H)$, then dimb(R_k,v) = $(b - \{(S_i,O_j,\underline{x})\}, M,f,H)$. (-;38;-)

e, execute	the execute attribute in the set A of access attributes: it implies neither the ability to read the object nor the ability to alter it. (-;12,22;11)
E(H)	the set of edges implied by the hierarchy H; $\{(O_1,O_2), \epsilon\ O$ and $O_2 \epsilon\ H(O_1)\}$. (-;-;9)
error	a decision used to coordinate a set of rules. (-;13;-)
f, f*	a classification/category vector from the set F. (15;23;-)
F	$C^S \times C^O \times (PK)^S \times (PK)^O \times C^S \times (PK)^S$. (15;23;19)
$f_1, f_2, f_3, f_4, f_5, f_6$	components of a vector f from the set F. (15;23;19)
g	an element of RA denoting "give" or "grant." (-;22;-)
g_1	a partial function from $S \times V$ to C denoting the highest classification of any object currently accessed by a subject in write mode in a given state;

	$g_1(S,v) = \max\{f_2(O): (S,O,\underline{w}) \in b\}$. (-;-;14)
g_2	a partial function from $S \times V$ to PK denoting the smallest category set containing the category set of each object currently accessed by a subject in <u>write</u> mode in a given state; $g_2(S,v) = \cup\{f_4(O): (S,O,\underline{w}) \in b\}$. (-;-;14)
G(H)	the digraph canonically generated from a hierarchy H; $G(H) = (O, E(H))$. (-;-;9)
H	a hierarchy from the set H of hierarchies. (-;-;8)
H	the set of hierarchies; $H \in H \subseteq (PO)^{I}$ if (1) $O_1 \neq O_2 \Rightarrow H(O_1) \cap H(O_2) = \phi$ and (2) there does not exist a set $\{O_1,\ldots,O_w\}$ such that $O_{r+1} \in H(O_r)$ where $1 \leq r \leq w$ and $O_{w+1} = O_1$. (-;-;8)
H_M	the subset of H consisting of hierarchies H with the vertices of its tree being precisely the active

	objects; {H ∈ H: $H^{-1}(\phi) - \cup H(O) = \{O_j: j \notin A(M)\}$}. (-;-;10)
h_1	a partial function from S × V to C denoting the highest classification of any object currently accessed by a subject in <u>read</u> mode in a given state; $h_1(S,v) = \max\{f_2(O): (S,O,\underline{r}) \in b\}$. (-;-;14)
h_2	a partial function from S × V to PK denoting the smallest category set containing the category set of each object currently accessed by a subject in <u>read</u> mode in a given state; $h_2(S,v) = \cup\{f_4(O): (S,O,\underline{r}) \in b\}$. (-;-;14)
<u>illegal</u>	a decision used to eliminate the <u>?</u> decision and to make a set of rules covering. (-;13;-)
K	the category set. (14;22;-)
K_r	a category from the set K. (14;22;-)
M,M^*,M_k	an access matrix from the set of all access matrices; an n × m matrix with entries from PA. (16;24;-)

no	a decision from the set D. (-;13;-)
O, O_j, O_k, O_1, O_2	an arbitrary object from the set $\mathcal{O}$ of objects. (14;22;-)
$\mathcal{O}$	the set of objects. (14;22;-)
P_α	the power set of P_α; the set of all subsets of α. (15;-;-)
Q	an arbitrary category set contained in K. (-;-;33)
r	an element of RA denoting "release" or "rescind." (-;22;-)
r, read	the see-only attribute in the set A of access attributes. (-;12,22;11)
R	the set of requests; in Volume III, the disjoint union of the sets $R^{(1)}$, $R^{(2)}$, $R^{(3)}$, $R^{(4)}$, and $R^{(5)}$. (15;22;-)
RA	request elements; {g, r, c, d} in Volume II and {g, r} in Volume III. (-;22;12)
R_k	an arbitrary request from the set R. (15;22;-)
$R^{(1)}$	RA × S × O × A. (-;-;11)

$R^{(2)}$	$S \times RA \times S \times O \times$ ' (-;-,11)
$R^{(3)}$	$S \times O \times C \times PK \times X$. (-;-;11)
$R^{(4)}$	$S \times O$. (-;-;11)
$R^{(5)}$	$S \times C \times PK$. (-;-;11)
s(j)	the index of the object directly superior to a noninitial object in a hierarchy; $s(j) = \{k: O_j \in H(O_k)\}$. (-;-;43)
S, S_i, S_λ	an arbitrary subject from the set S. (14;22;-)
S	the set of all subjects. (14;22;-)
S'	a subset of S which represents the untrusted subjects; $S' \subseteq S$. (-;-;25)
S^+	the augmentation of S by the element ϕ; $S+ = S \cup \{\phi\}$. (-;22;-)
SC rel f	the security condition relative to f; a per-subject condition for security. (-;26;-)
T	the time-index set. (15;23;-)
t	an element of T; a time. (15;23;-)
U_{ρ_1}	sets whose vacuity imply the presence of the *-property; used in the statement of the rules in Volume II. (-;39;22)

v	an element of the set V of all states. (17;24;11)
V	the set of all states; $V \subseteq P(S \times O \times A) \times M \times F \times H$, $H \in H_M$. (16;24;11)
w, <u>write</u>	the see-and-alter attribute in the set A of access attributes. (-;12,22;11)
$W(\omega)$	the relation generated by a set ω of rules. (-;28;-)
x	a request sequence from the set X. (16;23;-)
<u>x</u>	an arbitrary access attribute from the set A. (-;26;-)
X	the set of request sequences: R^T. (16;23;-)
x_t	the t-th request in the sequence x. (16;23;-)
y	a decision sequence from the set Y. (16;23;-)
Y	the set of decision sequences; D^T. (16;23;-)
<u>yes</u>	a decision from the set D. (-;13;-)
y_t	the t-th decision in the sequence y. (16;23;-)

z	a state sequence from the set Z. (16;24;-)
Z	the set of state sequences; V^T. (16;24;-)
z_0	an initial state. (17;25;-)
z_t	the t-th state in the sequence z. (16;24;-)

Greek Alphabet List

α_1	α_1: $O \times F \times C \times PK \rightarrow F$; α_1 alters the images of O_j under f to C_u and Q, respectively; $\alpha_1(O_j,f,C_u,Q) = f^* \in F$ where $f_t^* = f_t$ for t = 1, 3, 5, or 6; $f_2^*(O) = f_2(O)$ if $O \neq O_j$ and $f_2^*(O_j) = C_u$; and $f_4^*(O) = f_4(O)$ if $O \neq O_j$ and $f_4^*(O_j) = Q$. (-;-;33)
α_2	α_2: $S \times F \times C \times PK \rightarrow F$ by $\alpha_2(S_i,f,C_u,Q) = f^{**} \in F$ where $f_t^{**} = f_t$ for t = 1, 2, 3, and 4; $f_5^{**}(S) = f_5(S)$ if $S \neq S_i$ and $f_5^{**}(S_i) = C_u$; $f_6^{**}(S) = f_6(S)$ if $S \neq S_i$ and $f_6^{**}(S_i) = Q$. (-;-;50)

β_1	$\beta_1: O \times H \times M \rightarrow H$; $\beta_1(O_j,H,M) = H^* \in H$ where $H^*(O) = H(O)$ if $O \neq O_j$ and $H^*(O_j) = H(O_j) \cup \{O_{\tau(j,M)}\}$; β_1 attaches the "first" inactive object to O_j in the hierarchy H. (-;-;33)
β_2	$\beta_2: O \times H \rightarrow H$; $\beta_2(O_j,H) = H^* \in H$ where $H^*(O) = \begin{cases} H(O) - \{O_j\} & \text{if } O = O_{s(j)} \\ \phi & \text{if there is a set } \{O_1, \ldots, O_w\} \text{ with } O_{i+1} \in H(O_i) \text{ for } 1 \leq i < w,\ O_j = O_1, \text{ and } O = O_w \\ H(O) & \text{otherwise;} \end{cases}$ ζ_2 removes the subtree rooted at O_j from H. (-;-;47)
γ	an arbitrary element of RA. (-;38;-)
Δ_1	$\Delta_1(j,H) = \{k$: there exist $1 \leq u_0, \ldots, u_w \leq m$ where $u_0 = j$, $u_w = k$ and $O_{u_{i+1}} \in H(O_{u_i})$ for $0 \leq i < w\}$. (-;-;66)

Δ_2	$\Delta_2(j,b) = b - \{(S_i, O_k, \underline{x})$: $1 \leq i \leq n$, $k \in \Delta_1(j,H)$, $\underline{x} \in A\}$. (-;-;47)
Δ_3	$\Delta_3(j,M) = M \Theta [\underline{r},\underline{e},\underline{w},\underline{a}]_{1 \leq \lambda \leq n, k \in \Delta_1(j,H)}$.
Θ	See the Symbol List.
ρ, ρ_i	a rule; ρ: $R \times V \rightarrow D \times V$. (-;27;-)
σ_1, σ_2	arbitrary elements of S^+ in Volume II. (-;38;-)
σ_i	*-property functions in Volume III. (-;-;37)
$\Sigma(R,D,W,z_0)$	the system under investigation; $\Sigma(R,D,W,z_0) \subseteq X \times Y \times Z$ with $(x,y,z) \in \Sigma(R,D,W,z_0)$ if $(x_t,y_t,z_t,z_{t-1}) \in W$ for each $t \in T$. (17;25;-)
$\tau(j,M)$	a function to identify a unique inactive object index; for specificity, $\tau(j,M)$ was defined to be $\min\{k: j < k \leq m$ and $k \notin A(M)\}$. (-;-;33)
ϕ, φ	See the Symbol List.
Φ	a subset of A. (-;38;45)
χ	an arbitrary element of $\mathcal{X}$. (-;-;48)

χ	$A \cup \{\phi\} \cup F$ in Volume II; $\{\underline{e}, \phi\}$ in Volume III. (-;22;12)
ω	an arbitrary set of rules or the set $\{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, \rho_7, \rho_8, \rho_9, \rho_{10}\}$ in Volume II. (-;28, 51;-)
ω_{iii}	a set of rules for distributed implicit control; Volume III notation for the set $\{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_{12}, \rho_{13}, \rho_{14}, \rho_{15}, \rho_{17}\}$. (-;-;13)
ω'_{iii}	a set of rules for distributed implicit control and for the maintenance of compatability; differs from ω_{iii} by the substitution of ρ_{16} for ρ_{14}; $\{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_{12}, \rho_{13}, \rho_{15}, \rho_{16}, \rho_{17}\}$. (-;-;34)

Symbol List

*-property	a property of a state which does not allow the possibility of improper mixing of classified information by any subject; a Volume II concept which is replaced in Volume III by "*-property relative to S'." (-;28;-)
*-property-preserving	description of a rule that generates

	a state satisfying *-property (relative to some S') from any state satisfying the same property. (-;28;-)
*-property relative to S'	a property of a state which does not allow the possibility of improper mixing of classified information by any subject in the set S'; the Volume III replacement for "*-property." (-;-;25)
?	a decision used to coordinate a set of rules. (-;13;-)
ϕ, φ	the empty set
$\lhd_f$	the ordering of objects implicit in the functions f_2 and f_4; $O_1 \lhd_f O_2 \iff f_2(O_1) \leq f_2(O_2)$ and $f_4(O_1) \subseteq f_4(O_2)$. (-;-;28)
⊛	a symbol used in describing additions to an access matrix M; $M \circledast [\phi]_{ij}$ is the matrix M* where $M^*_{st} = \begin{cases} M_{st} & \text{if } (s,t) \neq (i,j) \\ M_{st} \cup \phi & \text{if } (s,t) = (i,j). \end{cases}$ (-;39;-)

Θ	a symbol used in describing deletions from an access matrix M; $M \circ [\Phi]_{ij}$ is the matrix M** where $M^{**}_{st} = \begin{cases} M_{st} & \text{if } (s,t) \neq (i,j) \\ M_{st} - \Phi & \text{if } (s,t) = (i,j). \end{cases}$ (-;39;-)

REFERENCES

1. D. Elliott Bell and Leonard J. La Padula, "Secure Computer Systems: Mathematical Foundations," MTR-2547, Vol. I, The MITRE Corporation, Bedford, Massachusetts, 1 March 1973.
2. Leonard J. La Padula and D. Elliott Bell, "Secure Computer Systems: A Mathematical Model," MTR-2547, Vol. II, The MITRE Corporation, Bedford, Massachusetts, 31 May 1973.
3. W. L. Schiller, "Design of a Security Kernel for the PDP-11/45," MTR-2709, The MITRE Corporation, Bedford, Massachusetts, 30 June 1973.
4. L. A. Smith, "Architectural Features for a Secure Computing System," (to be published).
5. K. G. Walter, W. F. Ogden, et. al., "Models for Secure Computer Systems," Interim Technical Report I, Department of Computing and Information Sciences, Case Western Reserve University, Cleveland, Ohio, 21 November 1973.
6. "Security Design Analysis Report for Air Force Data Services Center (AFDSC)," Honeywell Information Systems, Inc., December 1973.